개발 환경

- OS: Windows 10, 11

- IDE: VScode, Arduino 1.8.19, Processing

- Boards

+ STM nucleo-64 F103RB: <https://github.com/stm32duino/BoardManagerFiles/raw/main/package\_stmicroelectronics\_index.json>

+ NodeMCU 1.0: <https://arduino.esp8266.com/stable/package\_esp8266com\_index.json>

+ Arduino nano

- Library

+ RF24

+ Lidar TFmini: <https://github.com/Cesar-S/DFRobot\_TFmini>

+ MP3 DFPlayer: <https://github.com/DFRobot/DFRobotDFPlayerMini>

○ Software 설계도 (흐름도 및 클래스 다이어그램 등 / 개발언어에 따라 선택)

- // 너꺼에 해당하는 부분만 자세히 그려주면 됨

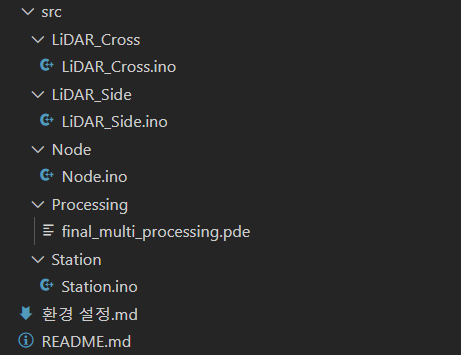
○ Software 기능 (필요 시 알고리즘 설명 포함)

- // 대충

○ 프로그램 사용법 (Interface)

- // 대충

파일 구성



소스 코드는 src 폴더 아래에 개발보드당 하나의 .ino 파일로 구성된다.

- LiDAR\_Cross: 횡단보도의 사람을 탐지하고 그 여부를 station으로 전송한다.

- LiDAR\_Side: 횡단보도 근처의 인도에 대해 사람을 탐지하고 그 여부를 station으로 전송한다.

- Node: 차량용 단말기에서 정보를 수신하고 이에 맞게 음성 안내를 출력한다.

- Station: LiDAR-nano slave에서 탐지 여부를 전송받아 우회전 가능/불능/경고를 구분하고 이를 단말기로 전송한다.

- 환경설정.md, README.md: 프로젝트의 진행과 환경 구성 진행 내용을 기록한다.

함수별 기능

- LiDAR

+ setup();

각종 프로토콜의 작동을 위한 설정을 한다. Serial, servo, TFmimi, Wire(I2C)의 begin을 하고 파라미터를 설정한다. I2C의 경우 기기끼리 서로 다른 slave 주소를 지정한다.

+ loop(); / Rotatemotor(); / CompareArray(int ia[][5]) - 이 부분은 나 말고 장한이나 민석이형이 앎

+ receiveFromMaster(int bytes);

I2C Master의 데이터 수신이 발생하면 그것을 송신하여 처리한다.

+ LiDAR: sendToMaster();

I2C Master의 request가 발생하면 주어진 flag 값을 전송한다.

- Node

+ setup();

각종 프로토콜의 작동을 위한 설정을 한다. MP3\_Serial, DFPlayer, radio(RF)의 begin을 하고 파라미터를 설정한다. RF의 경우 기기끼리 같은 주소를 지정한다.

+ loop();

RF 수신 메시지가 있을 경우 read buffer로 저장한다. Parsing 함수를 호출한 뒤 asserted된 flag에 따라 음성 안내 메시지를 출력한다.

+ nRF\_message\_parsing();

수신한 데이터를 미리 지정한 크기로 나누고 각 상태 변수에 저장한다. 또한 상태의 변화가 있을 시 이를 감지하여 해당 flag를 assert 한다.

- Station

+ setup();

각종 프로토콜의 작동을 위한 설정을 한다. Serial, Wire(I2C), radio(RF)의 begin을 하고 파라미터를 설정한다. RF의 경우 기기끼리 같은 주소를 지정한다. pwm으로 소리를 출력하기 위해 pinMode를 설정한다.

+ loop();

I2C를 통해 두 슬레이브 기기에게 데이터를 요청한다. 그것을 기반으로 RF 송신할 데이터를 만들고 전송한다.

이때 수신받을 단말기의 존재 여부를 Finite State Machine을 설계하여 이에 기반하여 알 수 있다. 먼저 singlecast로 ack를 받는 송신을 시작한다. 이때 ack가 돌아오지 않는다면 주변에 수신 가능한 단말기가 없는 것이므로 flag는 올라가지 않고 sinlgecast를 반복한다. 만약 ack가 돌아온다면 자동차 존재 여부 flag를 assert하고 타이머를 시작한다. 송신 모드 또한 broadcast로 전환한다. Broadcast 모드에서는 ack 여부를 확인하지 않고 주변의 모든 기기에게 데이터를 전송한다. 이 상태에서 타이머가 TIME\_BCT에 도달한다면 flag는 그대로 둔 채 singlecast로 전환한다. 만약 주변에 단말기가 아직 존재한다면 타이머를 리셋하고 바로 broadcast로 전활될 것이며 주변에 단말기가 없다면 계속 singlecast를 시도하다 TIME\_CAR 타이머가 끝날 때 flag를 de-assert한다.

flag에 따라 교차로에 설치된 중계기는 알람음을 출력한다. flag가 assert되어 있다면 주변에 통신가능한 단말기가 있으므로 차 또한 있다는 것이므로 스피커를 통해 반복되는 경고음을 출력한다. flag가 내려가면 종료한다.

+ I2C\_Req (int slaves);

특정 I2C 슬레이브에게 메시지 응답을 요청한다. 이때 전송해야 할 데이터의 크기 또한 지정한다. 전송받은 데이터는 변수에 저장해두어 RF 통신할 데이터를 만들 때 사용한다.

+ nRF\_make\_signal();

상기한 데이터를 기반으로 단말기에 전송해야 할 정보를 업데이트한다.

+ nRF\_make\_message();

비트 단위로 정보를 잘라 하나의 버퍼에 한 바이트 크기로 저장한다. 이 데이터는 loop에서 singlecast, broadcast할 때 전송된다.

- Processing

+ setup(); - 하승쓰 내용

○ 주요 함수의 흐름도

- //이것도 블럭 다이어 그램으로 그리면 됨

○ 기술적 차별성

- // 나머지는 노가단데 이거는 좀 생각해봐야하는 거니까 일단 대충 써놓으면 내가 정리하겠음

중계기